Problema 3.4: El receptari (proposta de solució)

Josep M. Ribó

19 de desembre de 2008

1 Objectius

- Dissenyar estructures de dades complexes

Per fer aquest problema, cal que us llegiu: • Els apunts del tema 3

2 `El receptari` (setembre-07

Un **plat** té un nom, uns aliments que li fan d'ingredients i una recepta per elaborar-lo (la recepta és simplement una cadena de caràcters).

Per la seva banda, un **ingredient** té un nom, una procedència i uns plats dels que forma part com a ingredients.

Volem dissenyar una classe `CatalegReceptes` que disposi de les operacions següents:

- `afegirNomPlat(...)`

  Afegeix al catàleg el nom d'un plat

- `afegirIngredient(...)`

  Afegeix un ingredient a un plat que ja s'ha introduït al catàleg (mitjançant l'operació afegirNomPlat(...)).

- `afegirRecepta(...)`

  Posa la recepta a un plat que ja s'ha introduït al catàleg (mitjançant l'operació `afegirNomPlat(...)`).

- `getInfoPlat(...)`

  A partir del nom d'un plat obté la seva informació associada (nom, recepta). Però no els seus ingredients (els ingredients s'obtenen amb una operació presentada més avall).

- `getPlatsPerIngredient(...)`

  Obté una llista de tots els plats als quals intervé un determinat ingredient

- `getIngredientsDePlat(...)`

  Obté una llista de tots els ingredients d'un determinat plat

- `getLlistaPlats(...)`

  Obté la llista dels plats del catàleg, ordenats per ordre alfabètic

Es demana:

1. (1 punt) Situa les classes `Plat, Ingredient` i `CatalegReceptes` a la jerarquia de classes definida a l'assignatura.

   Totes tres classes seran subclasses de `Object`.

2. Especifica la classe `CatalegReceptes`.

   En particular, només cal fer el següent:

   - (2 punts) Codificar en C++ la part pública de la classe `CatalegRecepetes`.
     En particular, et caldrà decidir quins paràmetres ha de tenir cadascuna de les operacions de la classe.
     Important: a les operacions `getPlatsPerIngredient`, `getIngredientsDePlat` i `getLlistaPlats` es vol que l'implementador de la classe tingui llibertat per canviar en un futur el tipus de llista que retornin aquestes operacions sense afectar als seus usuaris.

```
class CatalegReceptes :public Object{

 //.....REPRESENTACIO....

public:

 CatalegReceptes();

 void afegirNomPlat(const MyString& nomp);
        throw (ExcepcioPlatJaExistent);

 void afegirIngredient(const MyString& noming, const MyString& nomp);
         throw (ExcepcioPlatInexistent);

 void afegirRecepta(const MyString& nomp, const MyString& rec)
              throw (ExcepcioPlatInexistent);

 List<MyString>* getPlatsPerIngredient(const MyString& noming) const;

 void getInfoPlat(const MyString& nomp, Plat& plat, bool& found) const;

 List<MyString>* getIngredientsDePlat(const MyString& nomplat) const;

 List<MyString>* getLlistaPlats() const;
};
```

   La classe `Plat` es dissenyarà de la manera següent:

```
class Plat :public Object{

 MyString nomplat;
 MyString recepta;

public:

 Plat(){...}
 void setNomPlat(...){...}
 void getNomPlat(...){...}
 void setRecepta(...){...}
```

```
 void getRecepta(...){...}
....
//heretades d'Object
};
```

3. (2 punts) Especificar Pre-Post l'operació `afegirIngredient`.

   Considera la/les excepcions que hagi de llençar aquesta operació.

```
void afegirIngredient(const MyString& noming, const MyString& nomp);
        throw (ExcepcioPlatInexistent);
```

   - **Crida:** `cr.afegirIngredient(noming,nomp);`
   - **Pre:-**
   - **Post:** `cr` s'obté d'afegir al plat de nom `nomp` de `cr'` l'ingredient `noming`.
     Si `cr'` no conté cap plat anomenat `nomp` es llença l'excepció `ExcepcioPlatInexistent`.

4. Implementar la classe `CatalegReceptes`

   En particular, només cal fer el següent:

   - (3 punts) Representar la classe `CatalegReceptes` considerant que les operacions consultores han de ser eficients.

     Aplicarem el procediment per dissenyar la representació d'una classe complexa presentat al problema 3 (Diccionari):

     (a) *Determinar les operacions consultores de la classe.*
     - getInfoPlat
     - getPlatsPerIngredient
     - getIngredientsDePlat
     - getLlistaPlats()

     (b) *Per cadascuna de les operacions consultores obtingudes a 1, determinar la clau mitjançant la qual aquestes operacions accedeixen a la informació requerida*
     - getInfoPlat $\longrightarrow$ Nom del plat del qual es vol saber la informació (nom, recepta) (tipus de la clau: MyString)
     - getPlatsPerIngredient $\longrightarrow$ Nom de l'ingredient del que es vol conèixer en quins plats intervé (MyString)
     - getIngredientsDePlat $\longrightarrow$ Nom del plat del qual es vol conèixer quins ingredients el componen (MyString)
     - getLlistaPlats() $\longrightarrow$ En aquest cas no hi ha clau

     (c) *Dissenyar una taula per cadascuna de les claus trobades a 2*
     - `tinfoplats`

Clau	**Valor**	**Implementació**
Nom del plat (`MyString`)	Informació del plat(nom,recepta) (`Plat`)	BSTMap

     Comentari:

     * Proposem una implementació de la taula en forma de arbre binari de cerca (BSTMap) perquè la darrera operació consultora ha d'obtenir una llista de plats en ordre alfabètic (per nom de plat que és precisament la clau d'aquesta taula).

– `tingredients`

Clau	Valor	Implementació
Nom ingredient (`MyString`)	Llista noms plats que tenen l'ingredient (`LinkedList<MyString>`)	HashMap

Comentari:

∗ Com a implementació de la taula `tingredients` triem un `HashMap` perquè, en principi, proporciona una mica més d'eficiència, és més senzill gestionar possibles degeneracions i **no hi ha cap operació que requereixi obtenir els ingredients ordenats alfabèticament**.

– `tplats`

Clau	Valor	Implementació
Nom plat (`MyString`)	Llista noms d'ingredients del plat (`LinkedList<MyString>`)	BSTMap

COmentari:

∗ Proposem una implementació de la taula en forma de arbre binari de cerca (BSTMap) perquè la darrera operació consultora ha d'obtenir una llista de plats en ordre alfabètic (per nom de plat que és precisament la clau d'aquesta taula).

(d) *Si és possible, fusionar algunes de les taules del pas 3 per tal de minimitzar el nombre de taules de què constarà la representació de la classe*

La clau de les taules `tinfoplats` i `tplats` té el mateix significat conceptual (el nom d'un plat). Fàcilment podem crear una classe nova que fusioni la informació de la classe `Plat` i de la llista de noms d'ingredients. Quedaria així:

```
class PlatAmbIngre{
public:

  Plat plat;
  LinkedList<MyString> lingre;
};
```

Comentari:

– L'atribut `plat` no cal que sigui un apuntador a `Plat` perquè, en principi, la classe `Plat` no té subclasses i, per tant, no es requerirà un comportament polimòrfic, com sí que succeia en el cas de la classe `Word` al problema 3 (Dictionary).

– La classe `PlatAmbIngre` està dissenyada només per facilitar la implementació de `CatalegReceptes`. No la posarem a disposició del client. A més a més, és molt simple. Per tant, fem públics els seus atributs i hi accedim directament.

La taula resultat de la fusió de `tplats` i `tinfoplats` seria la següent:

– `tplatsAmbIngre`

Clau	Valor	Implementació
Nom del plat (`MyString`)	Info plat i llista ingredients (`PlatAmbIngre`)	BSTMap

Finalment, doncs, ens quedaran dues taules:

– `tingredients`

– `tPlatsAmbIngre`

(e) *Implementar la classe complexa en termes de les taules que han resultat d'aquest procés i de la resta d'atributs que siguin necessaris*

```
class  CatalegReceptes :public Object{

 BSTMap<WordInfoRel> tplatsAmbIngre;
 HashMap<LinkedList<MyString>, FH, BIngre> tingredients;

public:

//....

};
```

- (2 punts) Implementar l'operació `getIngredientsDePlat`.

```
List<MyString>* CatalegReceptes::getIngredientsDePlat
                          (const MyString& nomplat) const
{

   PlatAmbIngre infop;
   bool trobat;

   tplatsAmbIngre.get(nomplat,infop,trobat);

   if (trobat){

     return infop.lingre.clone();
   }
   else return new LinkedList<MyString>;

}
```